

# Image Enhancement in Spatial Domain

# Outline

- Grey Level Transformations
  - Negative, linear, piece-wise linear, log
- Histogram Processing
  - Equalization, matching (specification)
- Pseudo Color Enhancement
  - Density slicing, Pseudocolour transform
- Spatial Filtering

# Definition

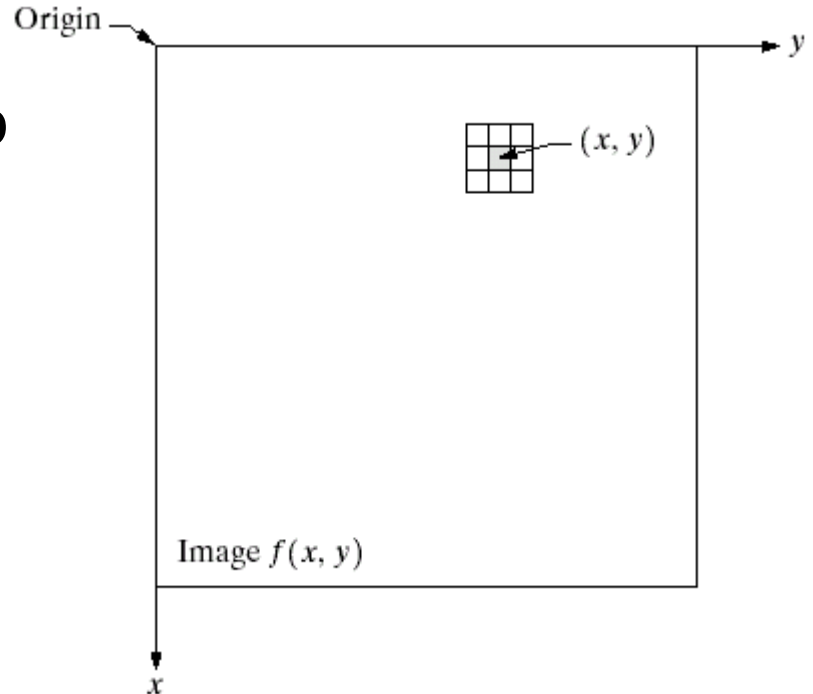
- Spatial Domain methods operate directly on pixels
- A process in Spatial Domain can be denoted by:

$$g(x, y) = T[f(x, y)]$$

- $f(x, y)$  is the input and  $g(x, y)$  is the output image
- $T$  is the operator (**Transformation function**) on  $f$ , defined over a neighborhood of  $(x, y)$
- A neighborhood over/about a pixel  $(x, y)$  uses a square/rectangular **sub-image** centered at  $(x, y)$

# 3x3 neighborhood of a pixel

- The center of **sub-image** is moved from the left top to the right bottom visiting all the pixels.
- $g(x,y)$  is calculated and **output image** is formed as each pixel is processed



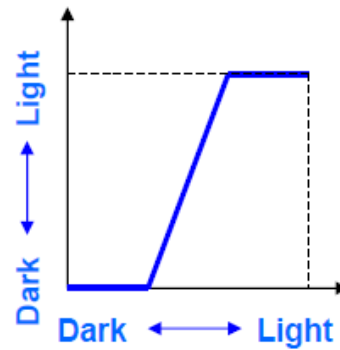
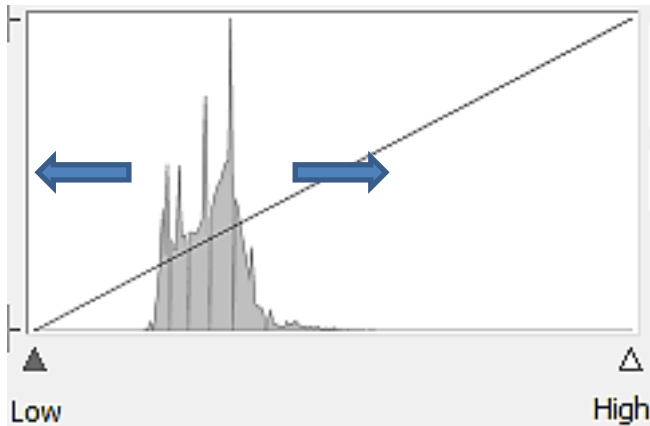
# Grey Level Transformation

- If a 1x1 neighborhood is used  $\Rightarrow T$  becomes a **gray-level** transformation function

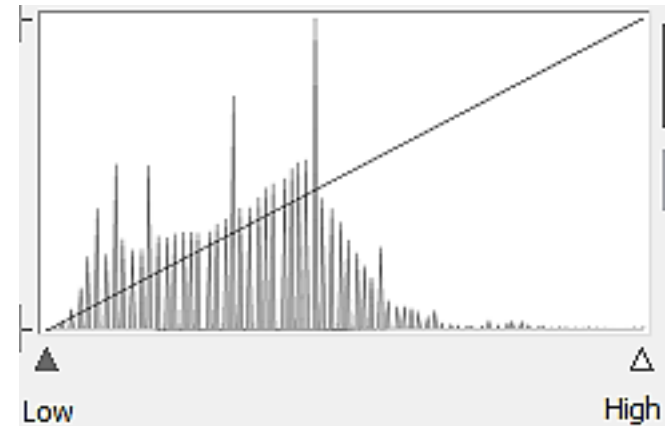
$$s = T(r), \text{ where, } s = g(x, y) \text{ and } r = f(x, y)$$

- Also called point processing
  - *Output pixel value **depends only** the grey level at that input point*
- Using a sub-image (mask or kernel)  $\Rightarrow$  **mask processing**

# Contrast Stretching

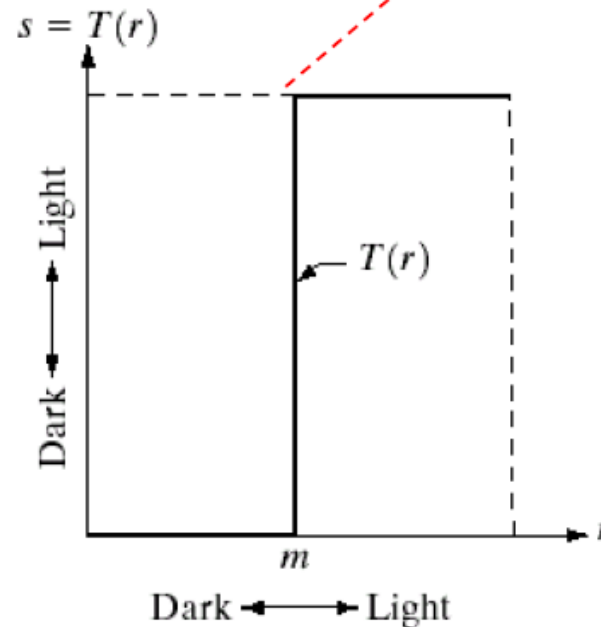
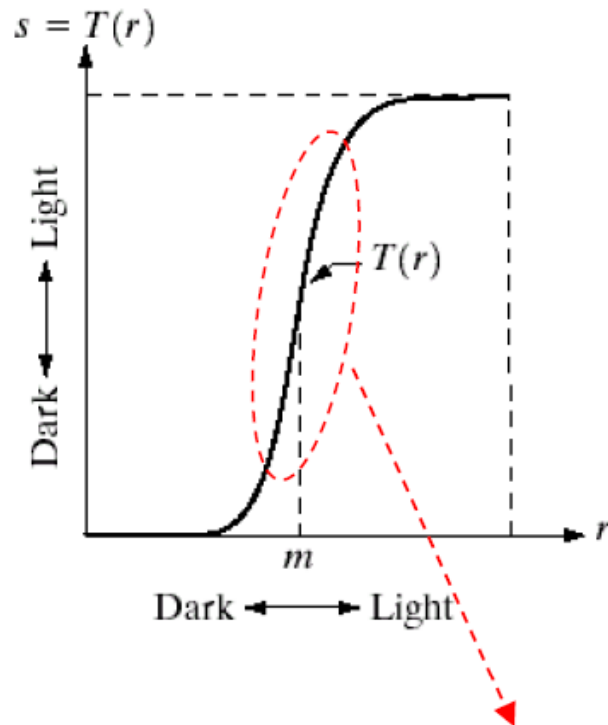


Linear Transformation  
(Scaling) Function



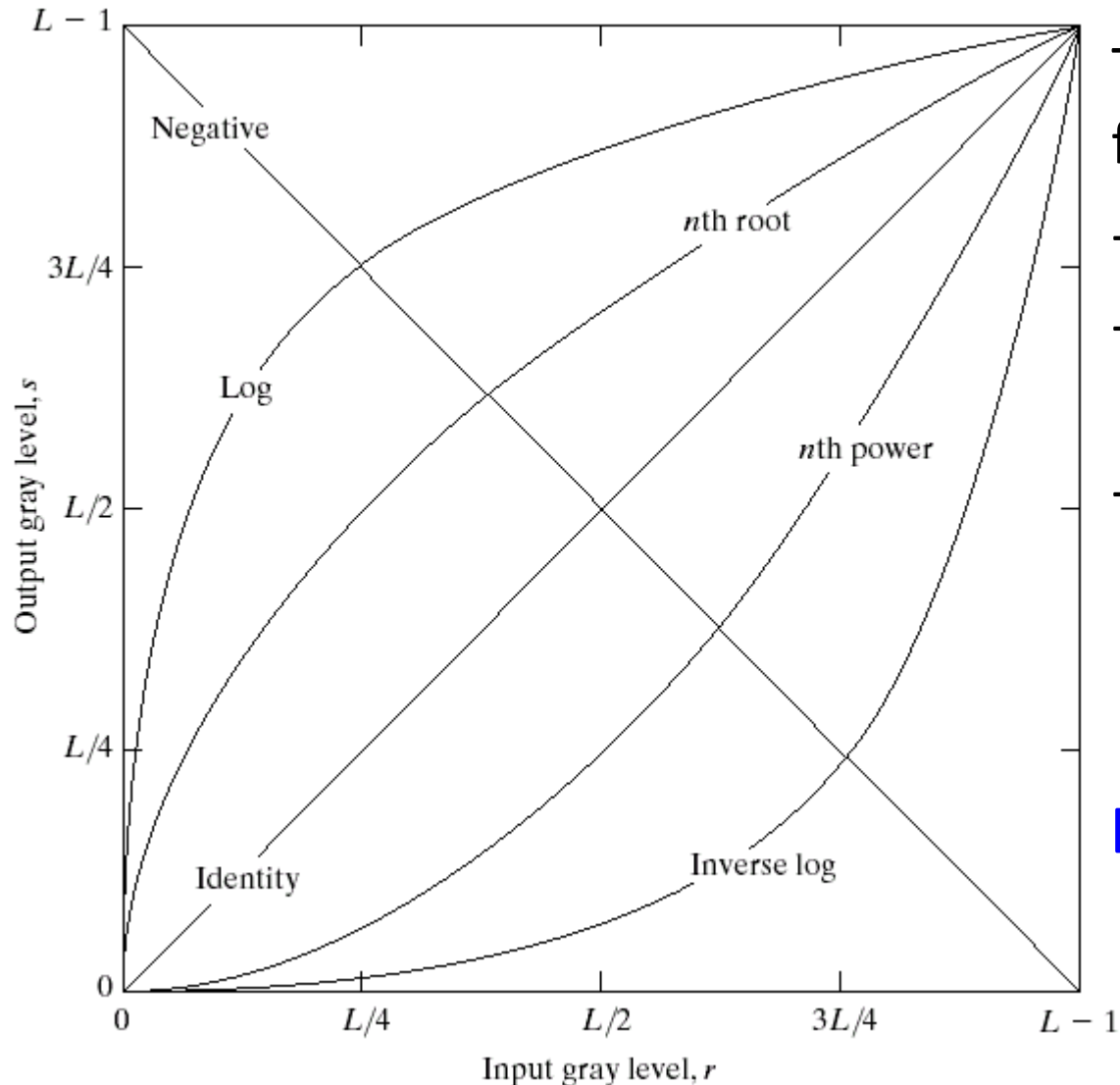
# Grey Level Transformation functions for Contrast Stretching

*Converts to black & white*



*Values  $< m$  are darkened, while  $> m$  are brightened non-linearly*

# Basic Gray Level Transformations



The transformation functions:

- linear
- logarithmic (log/inverse-log)
- power-law ( $n^{\text{th}}$  power/ $n^{\text{th}}$  root)

**[0, L-1]** is the gray-level range



# Image Negatives

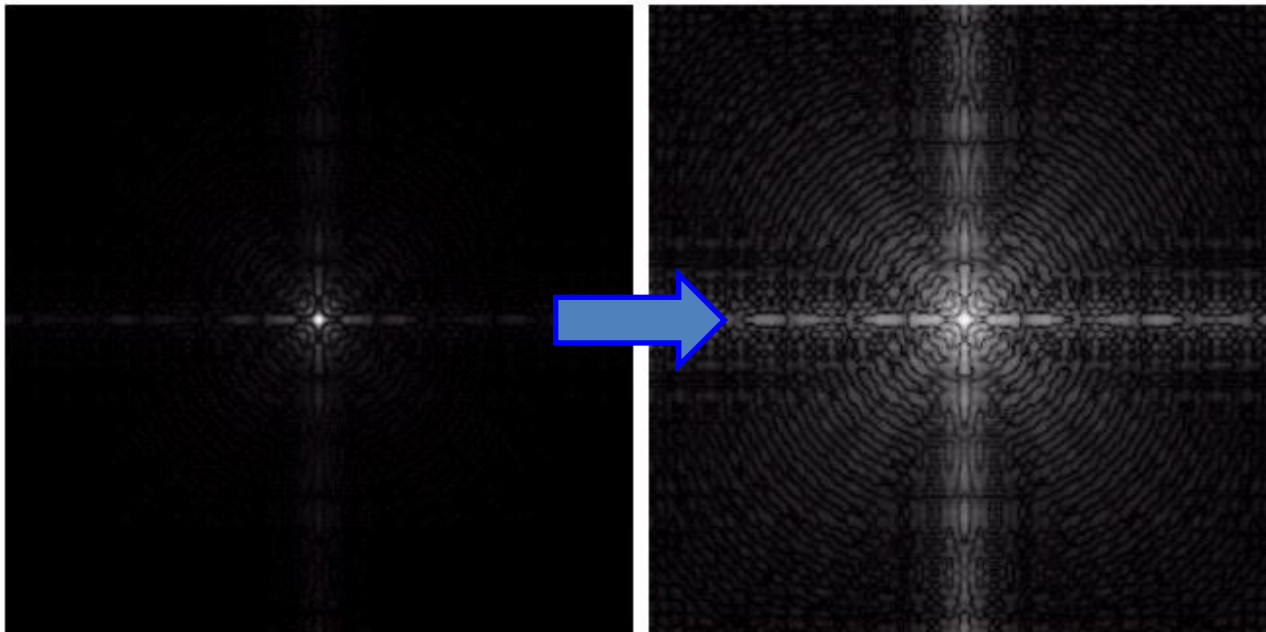
- Can be obtained by:  $S = L - 1 - r$



# Logarithmic Transformations

$s = c \log(1+r)$ ,  $c$  is a const. and  $r \geq 0$

- Log Transforms **compresses** the dynamic range of images with large variations in pixel values
- Typically, used to change the large dynamic range of a Fourier Spectrum to a smaller range for a **clearer inspection**

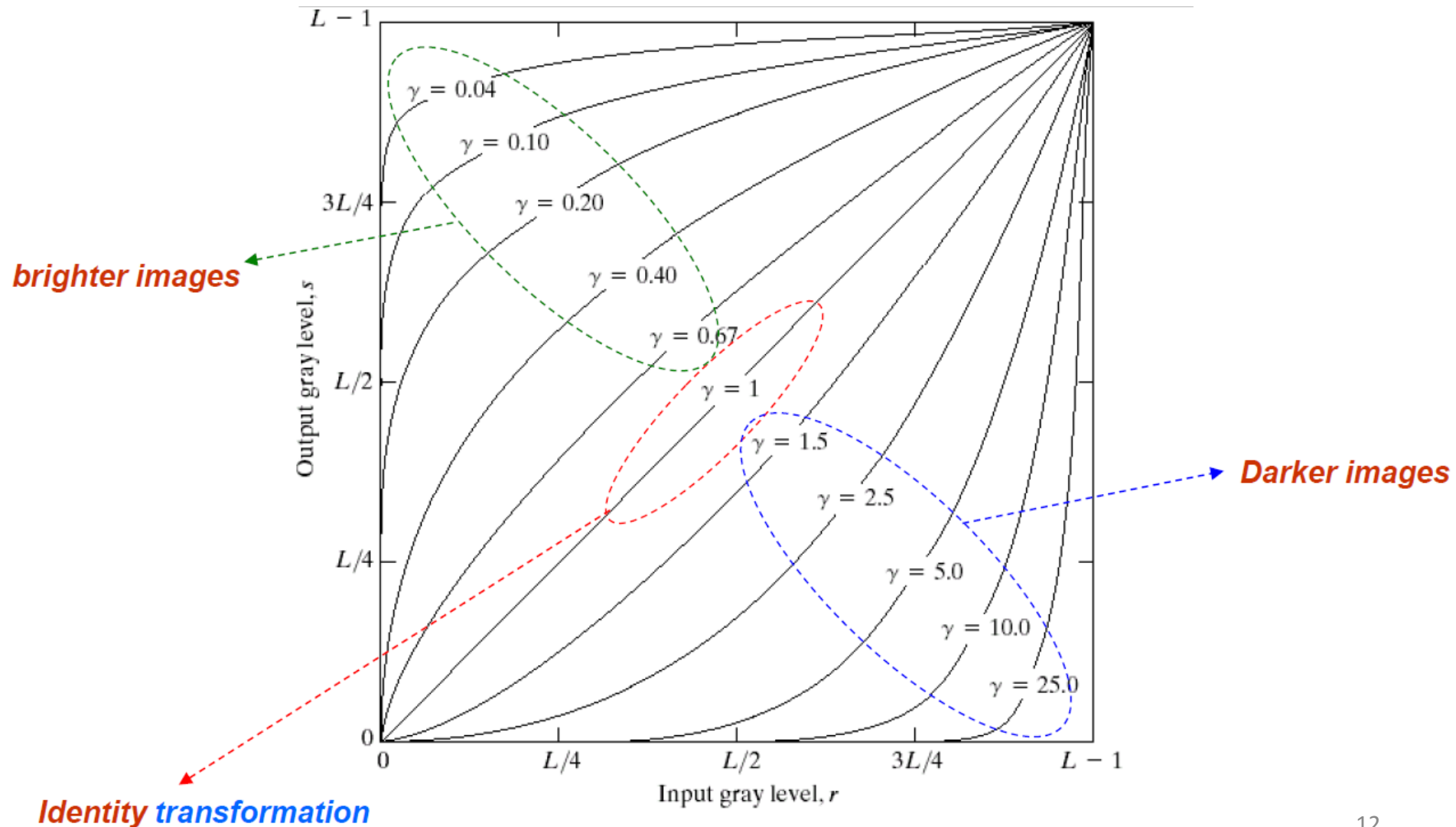


# Power-Law Transformations

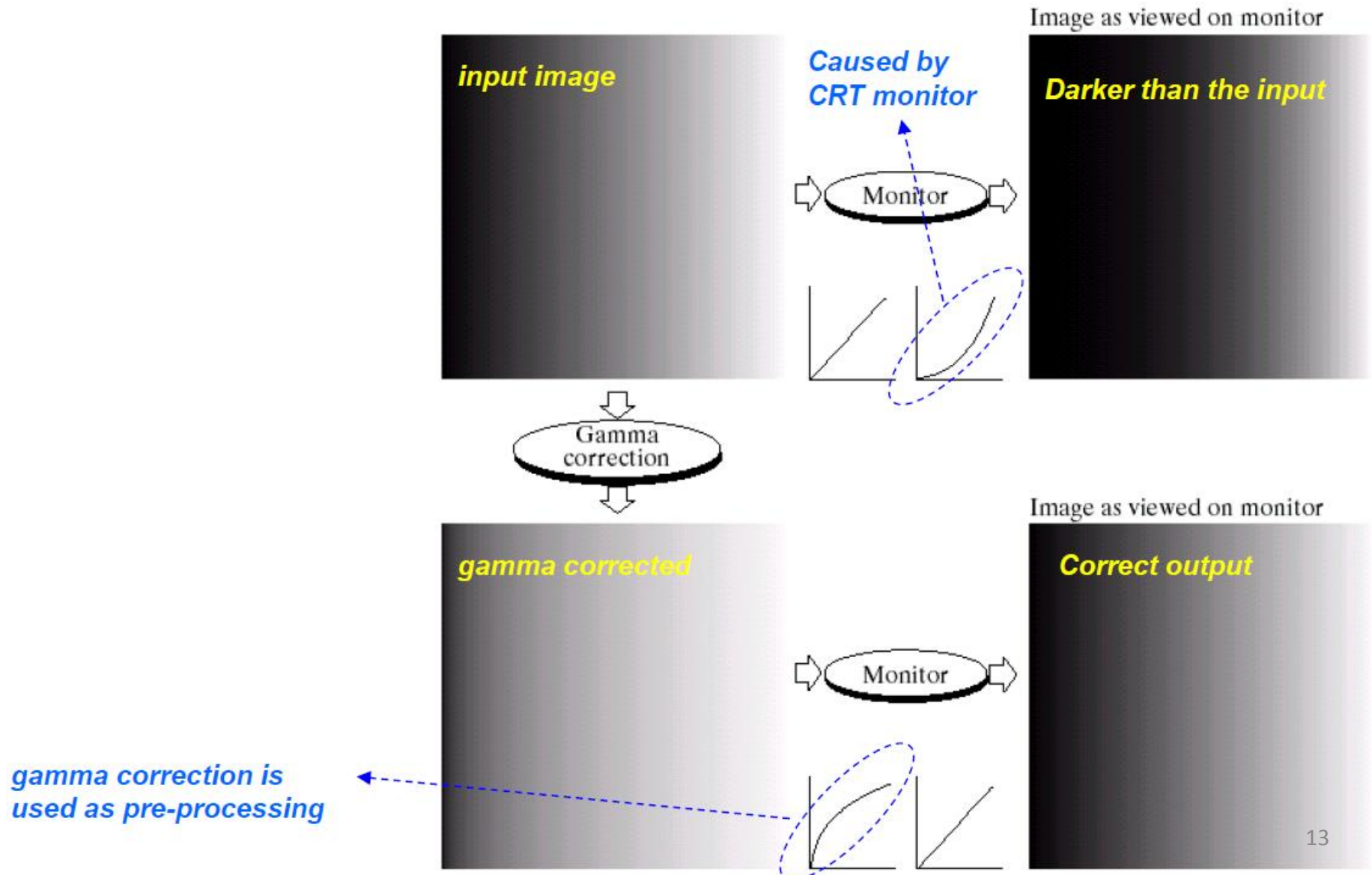
$s = c r^\gamma$ , where  $c$  and  $\gamma \geq 0$

- Different transformation curves are obtained by varying  $\gamma$  (gamma)
- Many image capturing, printing and display devices use gamma correction which enhances the given images by power-law response phenomena

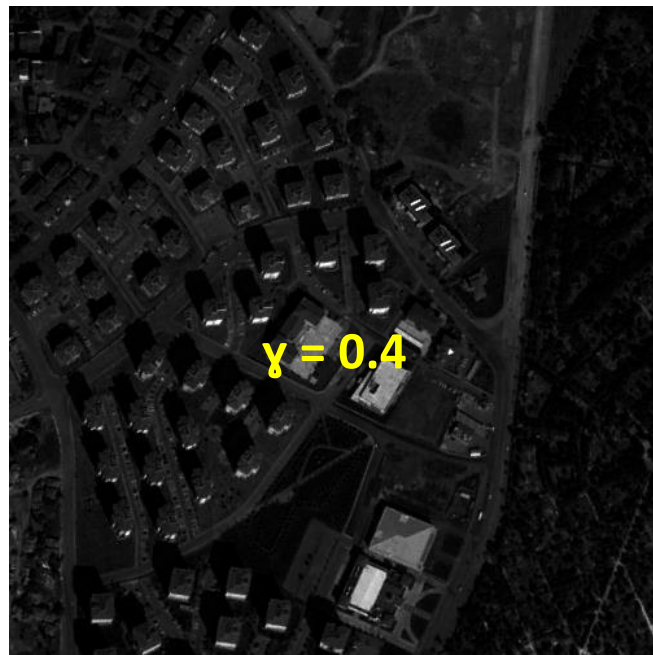
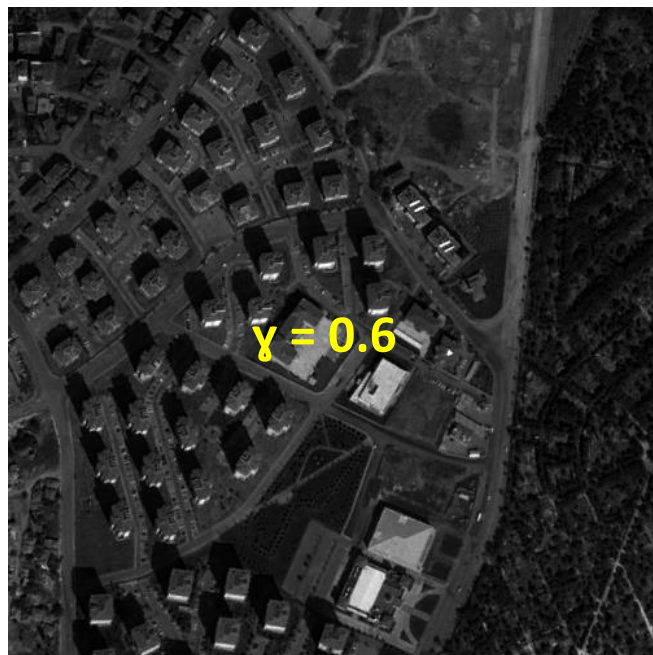
# Power-Law Transformations – Gamma Correction



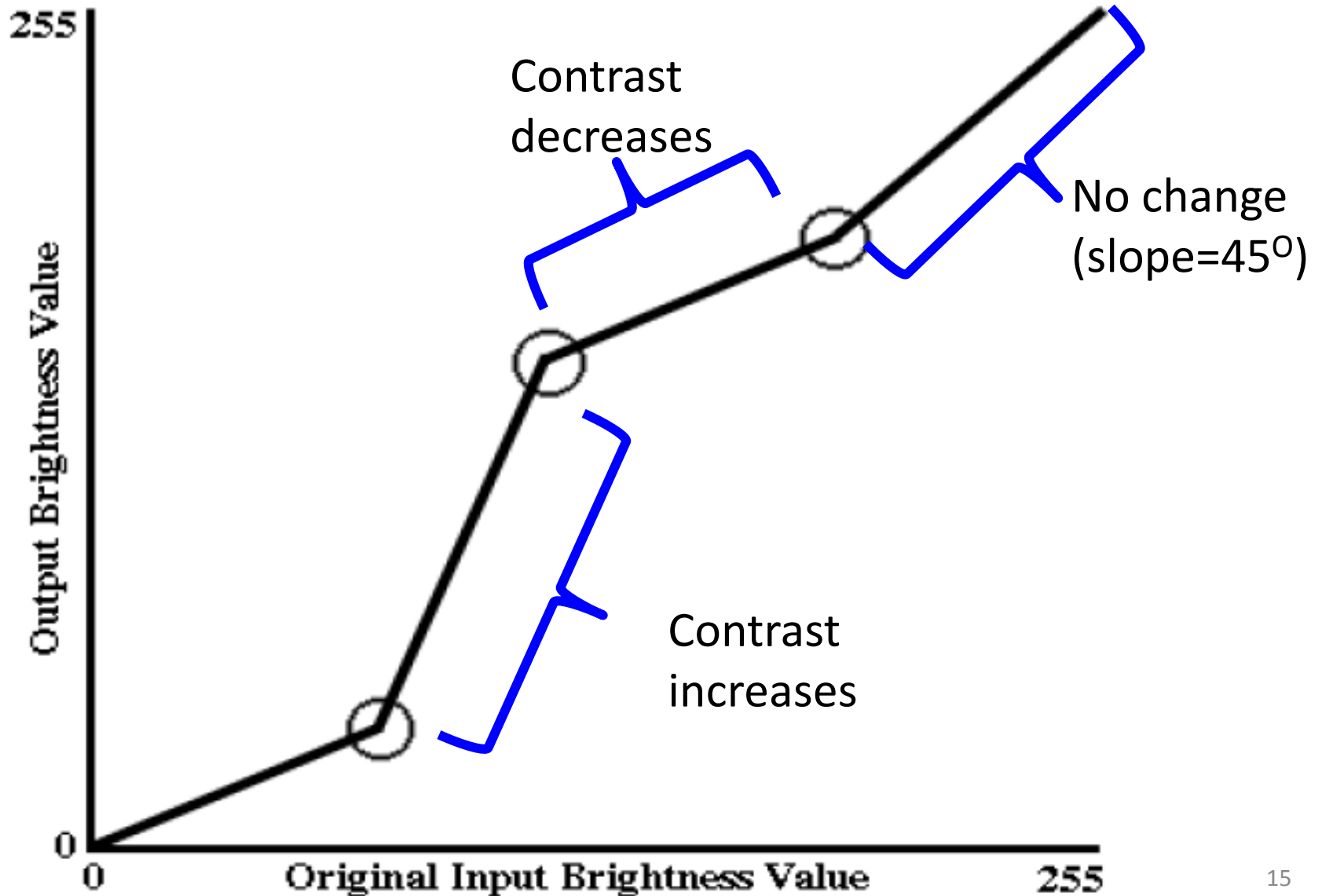
# Power-Law Transformations – Gamma Correction



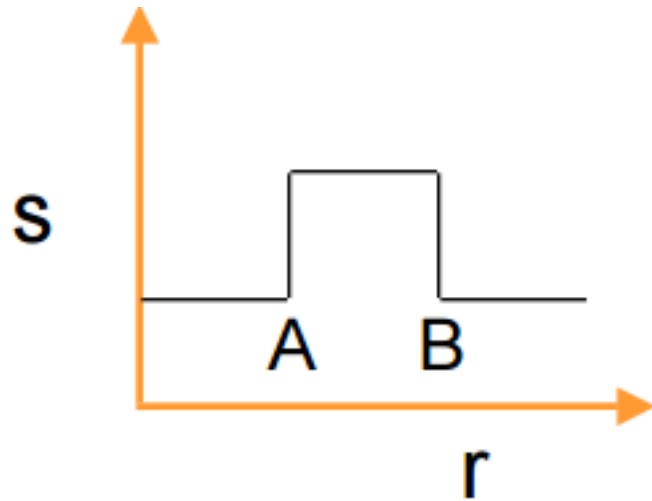




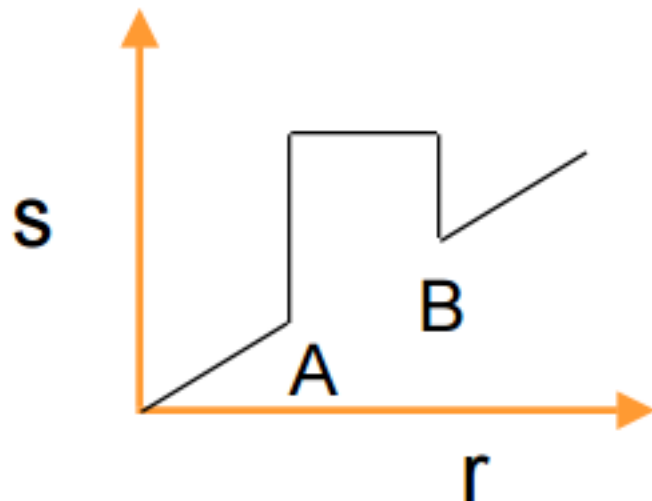
# Piecewise-Linear Transformation



# Gray-level Slicing



- Highlights only the range  $[A - B]$

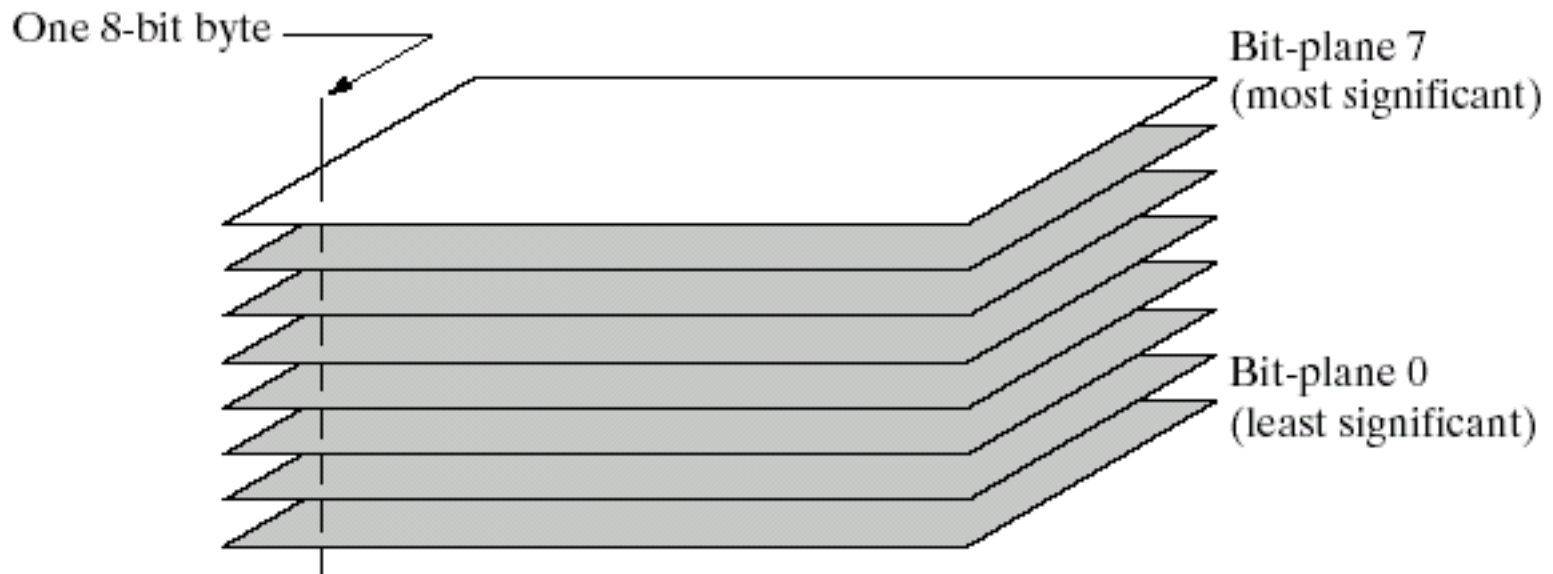


- Preserves other intensities



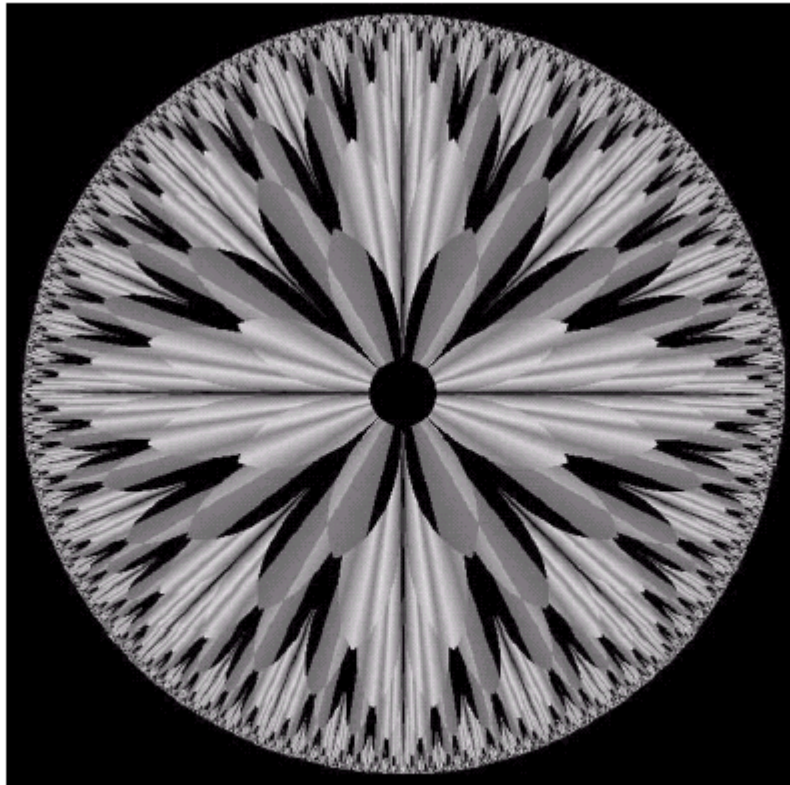
# Bit Plane Slicing

- Instead of highlighting gray-level ranges, the contribution made by each *bit* is highlighted
  - Used in *image compression*
  - Most significant bits contains the majority of the visually significant data



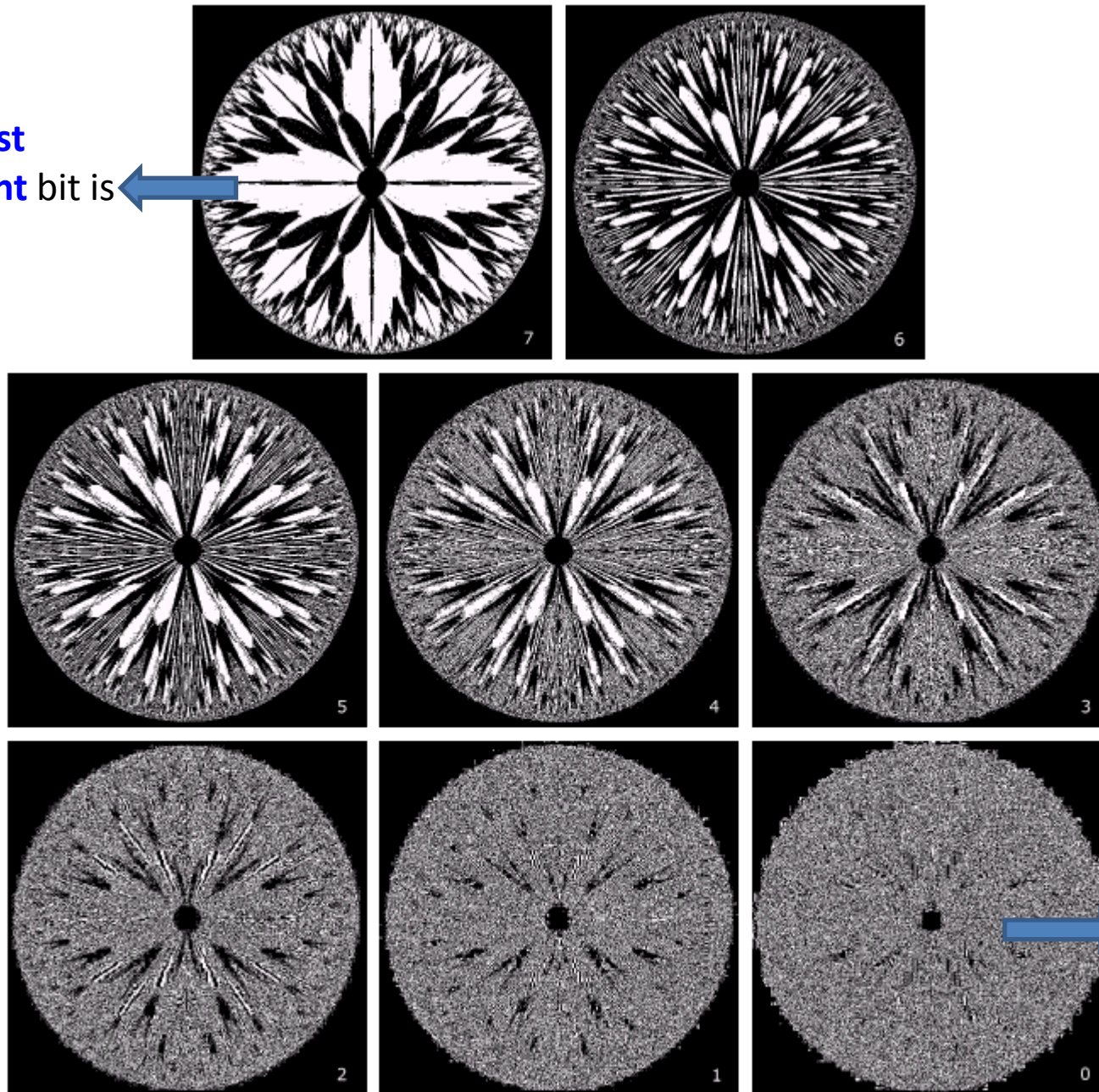
# Example

- An 8-bit fractal image



**FIGURE 3.13** An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

Only **most significant** bit is set to 1

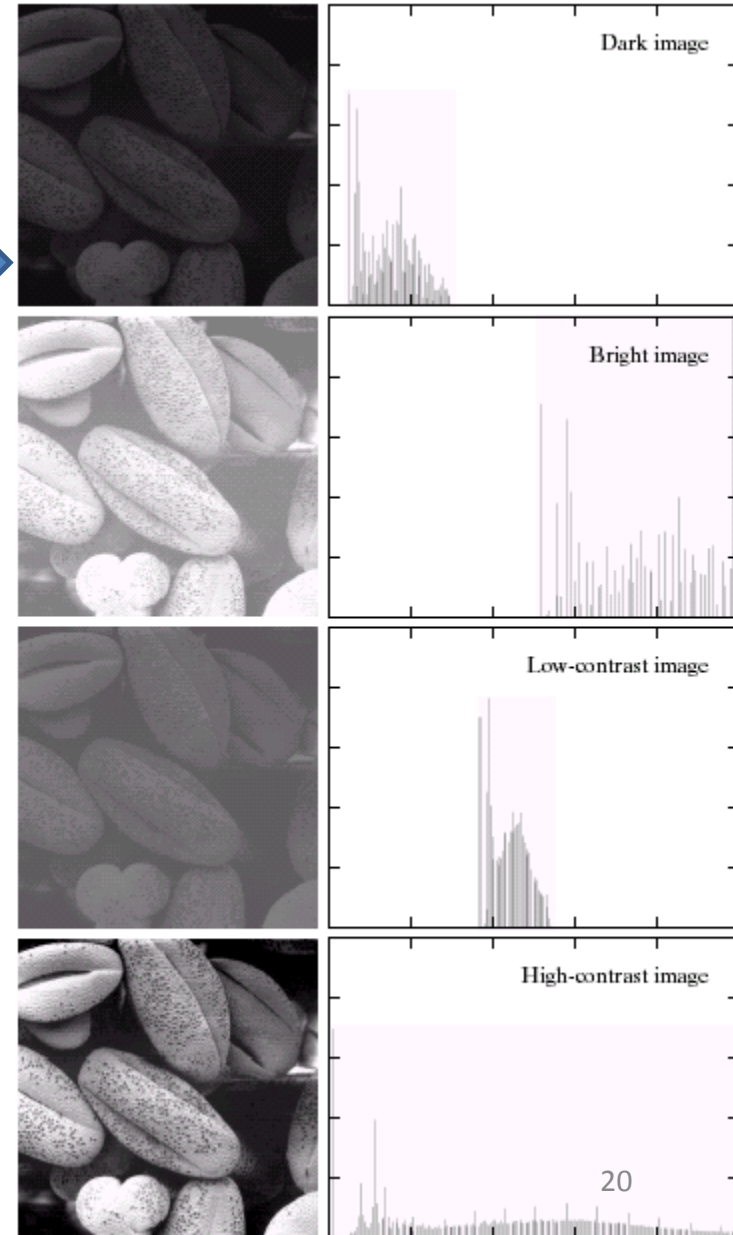


Only **least significant** bit is set to 1

**FIGURE 3.14** The eight bit planes of the image in Fig. 3.13. The number at the bottom, right of each image identifies the bit plane.

# Histogram Processing

- Histograms of four basic gray-level characteristics
- x axis: full range of the gray levels in the given image
- y axis: number of occurrences of each gray level value  $h(r_k)$   
or  
probability of a pixel to have that value,  $p(r_k)$





# Histogram Equalization

- Increases range of gray-levels in a low-contrast image to cover full range of gray-levels
- Achieved by a transformation function  $T(r)$ 
  - $T(r)$ : Cumulative Distribution Function (CDF) of the probability density function (PDF) of gray levels in the input image
  - Makes output PDF be uniform (not for discrete case)
- Input image has assumed to have continuous gray values ( $r$ ) normalized in the range  $[0,1]$

# Histogram Equalization

- For images with discrete gray values,

$$p_{\text{in}}(r_k) = \frac{n_k}{n}, \text{ for } 0 \leq r_k \leq 1, \text{ and } 0 \leq k \leq L-1$$

$L$ : Total number of gray levels

$n_k$ : Number of pixels with gray value  $r_k$

$n$ : Total number of pixels in the image

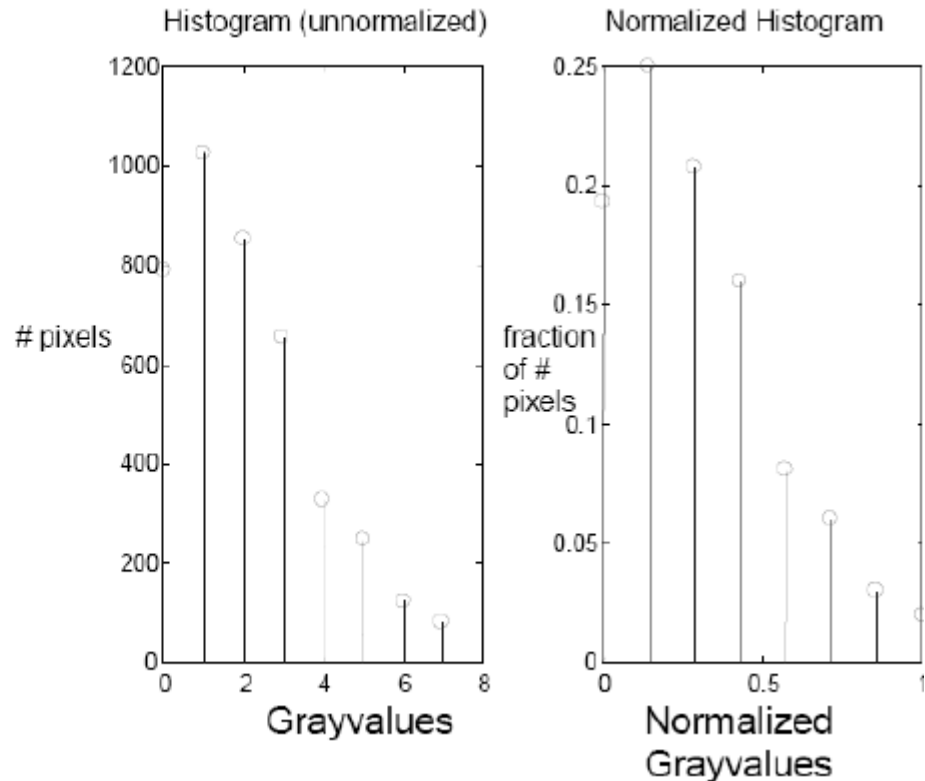
- Then, new value of  $r_k$  ( $s_k$ ) is:

$$s_k = T(r_k) = \sum_{j=0}^k \frac{n_j}{n} = \sum_{j=0}^k p_{\text{in}}(r_j), \text{ for } 0 \leq k \leq L-1$$

# Example

- Consider an 8-level 64 x 64 image with gray values (0, 1, ..., 7).
- Normalized gray values ( $r_k$ ) are (0, 1/7, 2/7, ..., 1)

$k$	$r_k$	$n_k$	$p(r_k) = n_k/n$
0	0	790	0.19
1	1/7	1023	0.25
2	2/7	850	0.21
3	3/7	656	0.16
4	4/7	329	0.08
5	5/7	245	0.06
6	6/7	122	0.03
7	1	81	0.02



# Example

Apply the transformation (round off to nearest gray level):

$$s_0 = T(r_0) = \sum_{j=0}^0 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) = 0.19 \rightarrow 1/7$$

$$s_1 = T(r_1) = \sum_{j=0}^1 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) = 0.44 \rightarrow 3/7$$

$$s_2 = T(r_2) = \sum_{j=0}^2 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + p_{\text{in}}(r_2) = 0.65 \rightarrow 5/7$$

$$s_3 = T(r_3) = \sum_{j=0}^3 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_3) = 0.81 \rightarrow 6/7$$

$$s_4 = T(r_4) = \sum_{j=0}^4 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_4) = 0.89 \rightarrow 6/7$$

$$s_5 = T(r_5) = \sum_{j=0}^5 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_5) = 0.95 \rightarrow 1$$

$$s_6 = T(r_6) = \sum_{j=0}^6 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_6) = 0.98 \rightarrow 1$$

$$s_7 = T(r_7) = \sum_{j=0}^7 p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_7) = 1.00 \rightarrow 1$$

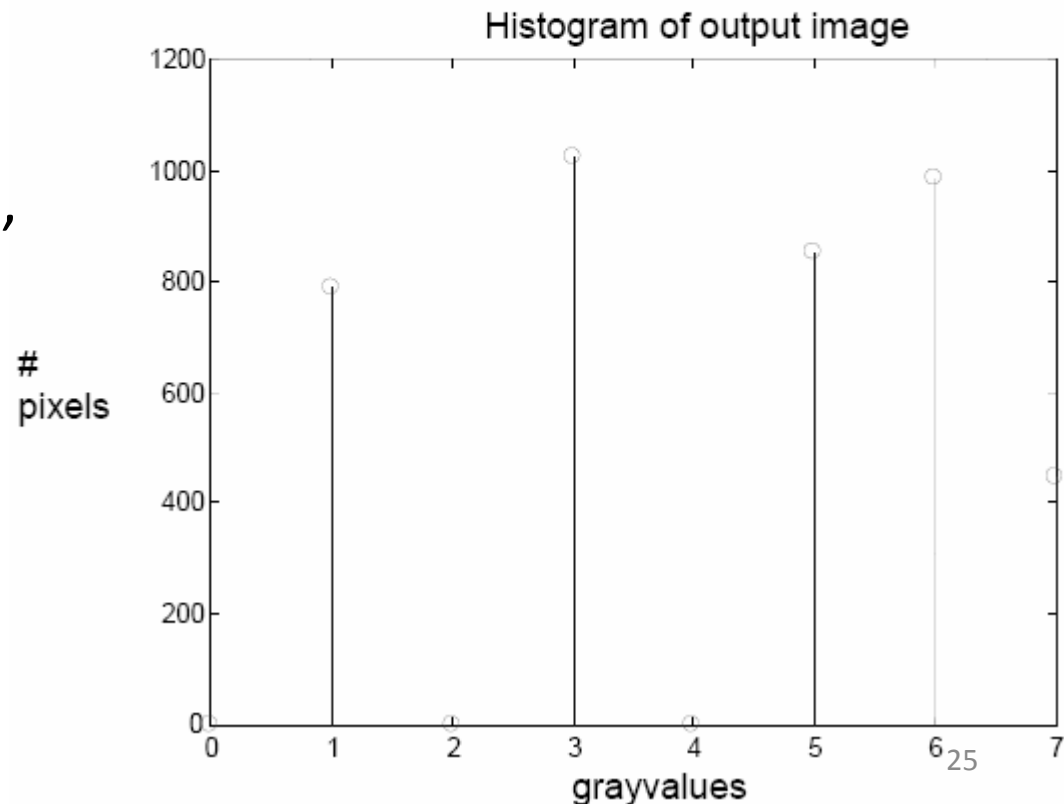


# Example

- After transformation, there are only 5 gray levels in output image:  $(1/7, 3/7, 5/7, 6/7, 1)$
- With this transformation, the output image will have histogram:

Almost uniform!!!

$k$	$s_k$	$n_k$	$p(s_k) = n_k/n$
0	$1/7$	790	0.19
1	$3/7$	1023	0.25
2	$5/7$	850	0.21
3	$6/7$	985	0.24
4	1	448	0.11



# Effect on a low-contrast image



# Histogram Specification (Matching)

- After Histogram equalization image pixels are (in theory) uniformly distributed
  - Sometimes, this may not be desirable
- Instead, we may want a transformation that yields an output image with a prespecified histogram
  - This technique is called histogram specification
  - Used in image de-stripping (Chapter 3)  
(in which stripe histograms are matched, not image hist.s)

# Histogram Specification (Matching)

- Idea:
  - $T(r)$  transforms *input* into a uniform histogram
  - $G(z)$  transforms *output* into a uniform histogram
  - Then apply following transformation to match input histogram to output histogram

$$z = G^{-1}[T(r)]$$

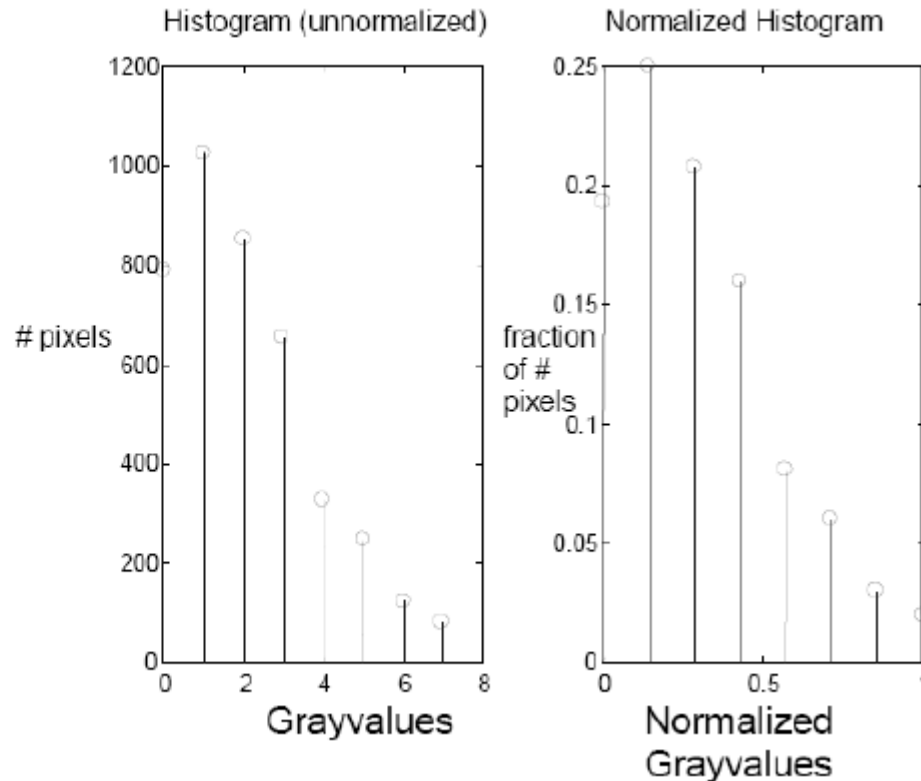
We already know how to transform into a uniform histogram:

- By *histogram equalization* (using CDF)

# Example

- Consider the previous 8-level 64 x 64 image

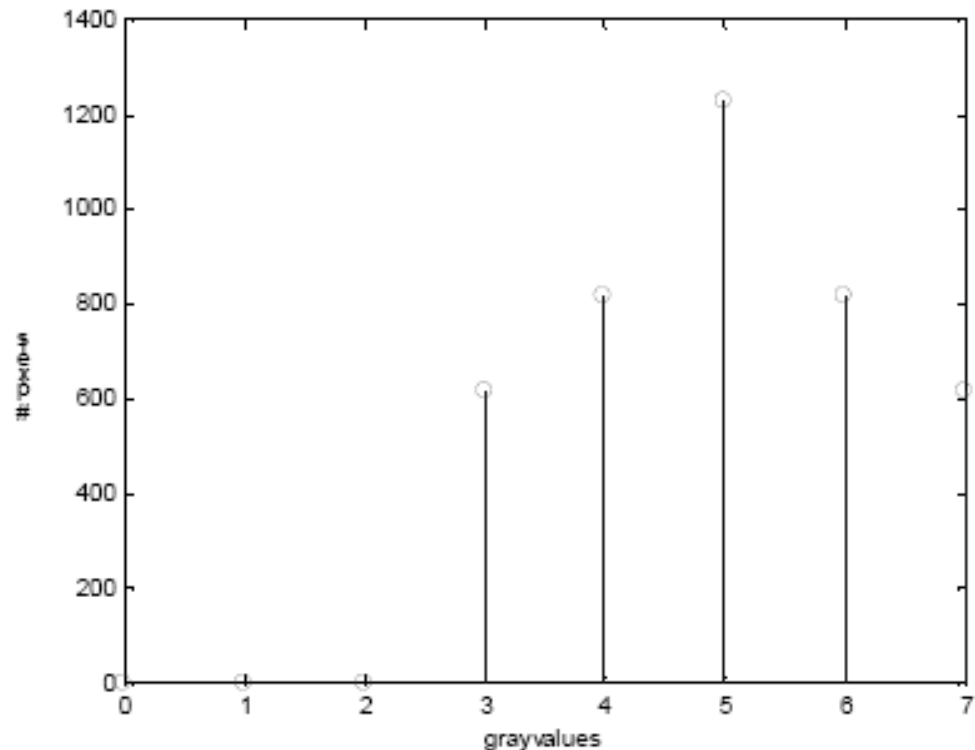
$k$	$r_k$	$n_k$	$p(r_k) = n_k/n$
0	0	790	0.19
1	1/7	1023	0.25
2	2/7	850	0.21
3	3/7	656	0.16
4	4/7	329	0.08
5	5/7	245	0.06
6	6/7	122	0.03
7	1	81	0.02



# Example

- We wish to transform this image into a new one which has the following histogram  $z = H(r) = G^{-1}[T(r)]$

$k$	$z_k$	$p_{\text{out}}(z_k)$
0	0	0.00
1	1/7	0.00
2	2/7	0.00
3	3/7	0.15
4	4/7	0.20
5	5/7	0.30
6	6/7	0.20
7	1	0.15



# Example

- In histogram equalization example, we have already obtained the transformation  $T(r)$ 
  - Using CDF
- So,  $T(r)$  is:

$r_j \rightarrow s_k$	$n_k$	$p(s_k)$
$r_0 \rightarrow s_0 = 1/7$	790	0.19
$r_1 \rightarrow s_1 = 3/7$	1023	0.25
$r_2 \rightarrow s_2 = 5/7$	850	0.21
$r_3, r_4 \rightarrow s_3 = 6/7$	985	0.24
$r_5, r_6, r_7 \rightarrow s_4 = 1$	448	0.11

# Example

Next:

Compute  $G(z)$   
from target  
histogram:

$k$	$z_k$	$p_{\text{out}}(z_k)$
0	0	0.00
1	1/7	0.00
2	2/7	0.00
3	3/7	0.15
4	4/7	0.20
5	5/7	0.30
6	6/7	0.20
7	1	0.15



$$v_0 = G(z_0) = \sum_{j=0}^0 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) = 0.00 \rightarrow 0$$

$$v_1 = G(z_1) = \sum_{j=0}^1 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) = 0.00 \rightarrow 0$$

$$v_2 = G(z_2) = \sum_{j=0}^2 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + p_{\text{out}}(z_2) = 0.00 \rightarrow 0$$

$$v_3 = G(z_3) = \sum_{j=0}^3 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \dots + p_{\text{out}}(z_3) = 0.15 \rightarrow 1/7$$

$$v_4 = G(z_4) = \sum_{j=0}^4 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \dots + p_{\text{out}}(z_4) = 0.35 \rightarrow 2/7$$

$$v_5 = G(z_5) = \sum_{j=0}^5 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \dots + p_{\text{out}}(z_5) = 0.65 \rightarrow 5/7$$

$$v_6 = G(z_6) = \sum_{j=0}^6 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \dots + p_{\text{out}}(z_6) = 0.85 \rightarrow 6/7$$

$$v_7 = G(z_7) = \sum_{j=0}^7 p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \dots + p_{\text{out}}(z_7) = 1.00 \rightarrow 1$$



# Example

$r_i \rightarrow s_k$	$n_k$	$p(s_k)$
$r_0 \rightarrow s_0 = 1/7$	790	0.19
$r_1 \rightarrow s_1 = 3/7$	1023	0.25
$r_2 \rightarrow s_2 = 5/7$	850	0.21
$r_3, r_4 \rightarrow s_3 = 6/7$	985	0.24
$r_5, r_6, r_7 \rightarrow s_4 = 1$	448	0.11

Notice that  $G$  is not invertible. But we will do our best:

$$G^{-1}(0) = ? \quad (\text{this does not matter, since } s \neq 0)$$

$$G^{-1}(1/7) = 3/7$$

$$G^{-1}(2/7) = 4/7 \quad (\text{this does not matter, since } s \neq 2/7)$$

$$G^{-1}(3/7) = 4/7 \quad (\text{this is not defined, but we use a close match})$$

$$G^{-1}(4/7) = ? \quad (\text{this does not matter, since } s \neq 4/7)$$

$$G^{-1}(5/7) = 5/7$$

$$G^{-1}(6/7) = 6/7$$

$$G^{-1}(1) = 1$$

# Example

Finally, combine the two transformations  $T$  and  $G^{-1}$  to obtain  $H$ :  $z = H(r) = G^{-1}[T(r)]$

$r \rightarrow T(r) = s$	$s \rightarrow G^{-1}(s) = z$	$r \rightarrow G^{-1}[T(r)] = H(r) = z$
$r_0 = 0 \rightarrow 1/7$	$0 \rightarrow ?$	$r_0 = 0 \rightarrow z_3 = 3/7$
$r_1 = 1/7 \rightarrow 3/7$	$1/7 \rightarrow 3/7$	$r_1 = 1/7 \rightarrow z_4 = 4/7$
$r_2 = 2/7 \rightarrow 5/7$	$2/7 \rightarrow 4/7$	$r_2 = 2/7 \rightarrow z_5 = 5/7$
$r_3 = 3/7 \rightarrow 6/7$	$3/7 \rightarrow 4/7$	$r_3 = 3/7 \rightarrow z_6 = 6/7$
$r_4 = 4/7 \rightarrow 6/7$	$4/7 \rightarrow ?$	$r_4 = 4/7 \rightarrow z_6 = 6/7$
$r_5 = 5/7 \rightarrow 1$	$5/7 \rightarrow 5/7$	$r_5 = 5/7 \rightarrow z_7 = 1$
$r_6 = 6/7 \rightarrow 1$	$6/7 \rightarrow 6/7$	$r_6 = 6/7 \rightarrow z_7 = 1$
$r_7 = 1 \rightarrow 1$	$1 \rightarrow 1$	$r_7 = 1 \rightarrow z_7 = 1$

# Pseudo Color Enhancement

- Human eye is more sensitive to changes in the color hue than in brightness
- Thus, this enhancement aims to use different colors to represent a gray level image
- 2 methods:
  - Density (Intensity) Slicing
  - Pseudo Coloring

# Density Slicing

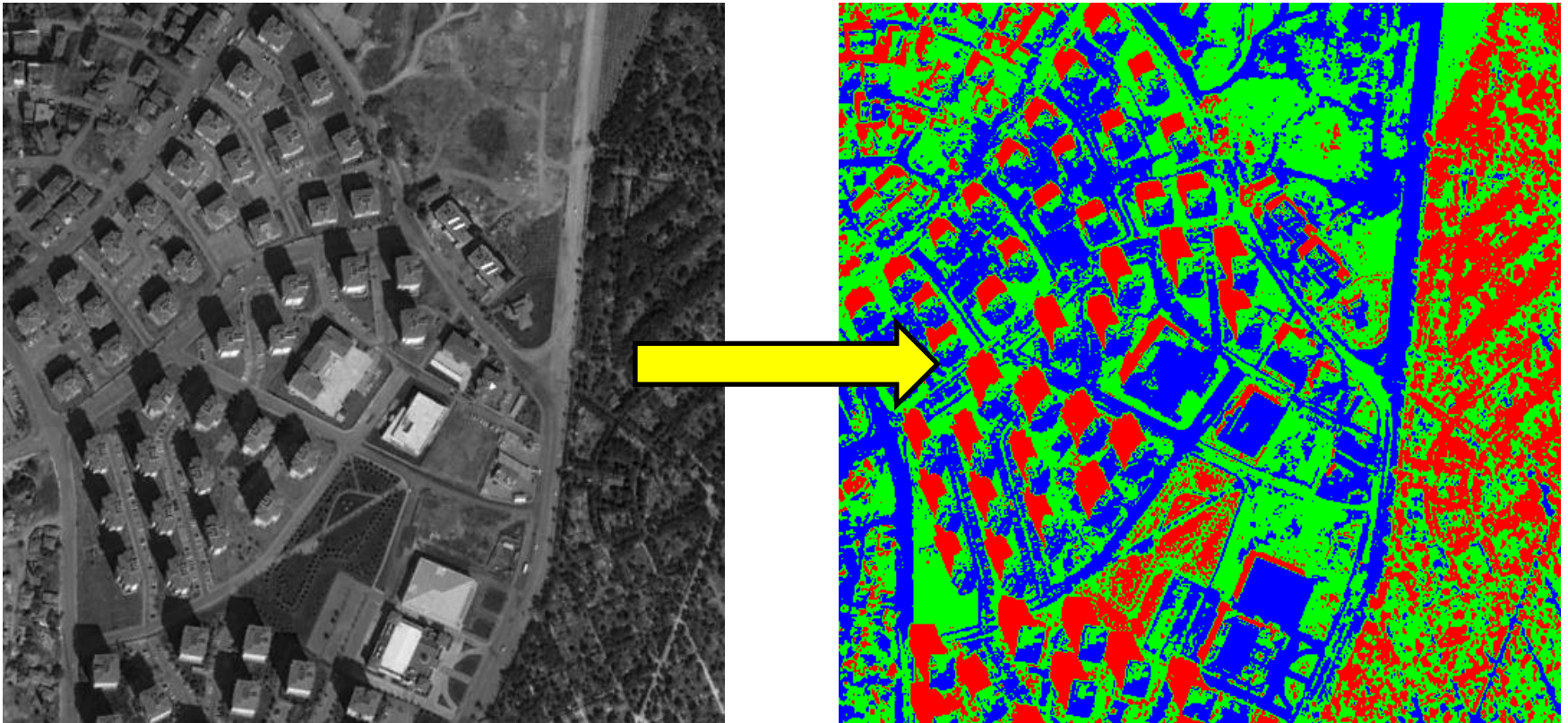
- Gray values are sliced into ranges
- A *color* is assigned to each range of gray values
- Used as selective one-dimensional classification
  - Where each color represents regions of a different *class*
- Slicing is done by choosing *multiple thresholds*

# Density Slicing

$[0,48]$  => red

$[48,96]$  => green

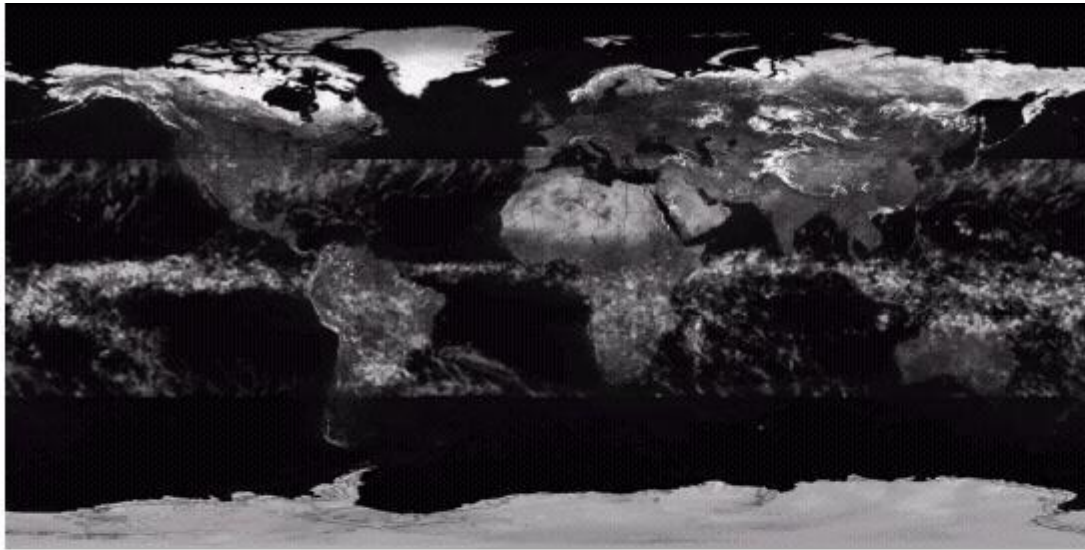
$[96,255]$  => blue



# Pseudo Coloring

- A technique to represent all grey levels using different colors
  - Done by using LUTs (Look Up Tables)
  - LUT: A table showing which grey value will be mapped to which color
- This has been used in coloring classification maps in most image analysis software systems

# Example: Pseudo Coloring

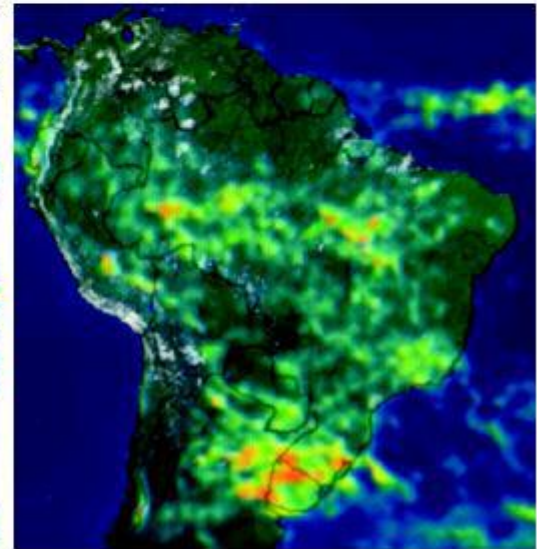
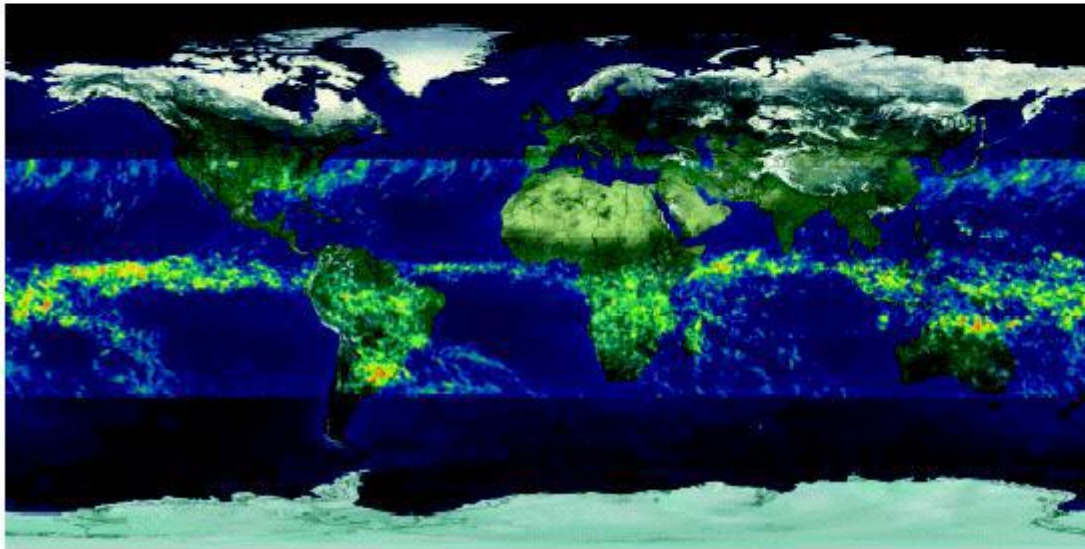
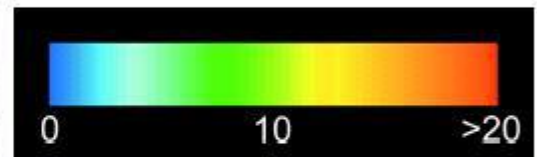


LUT for 256 gray levels:

0



255

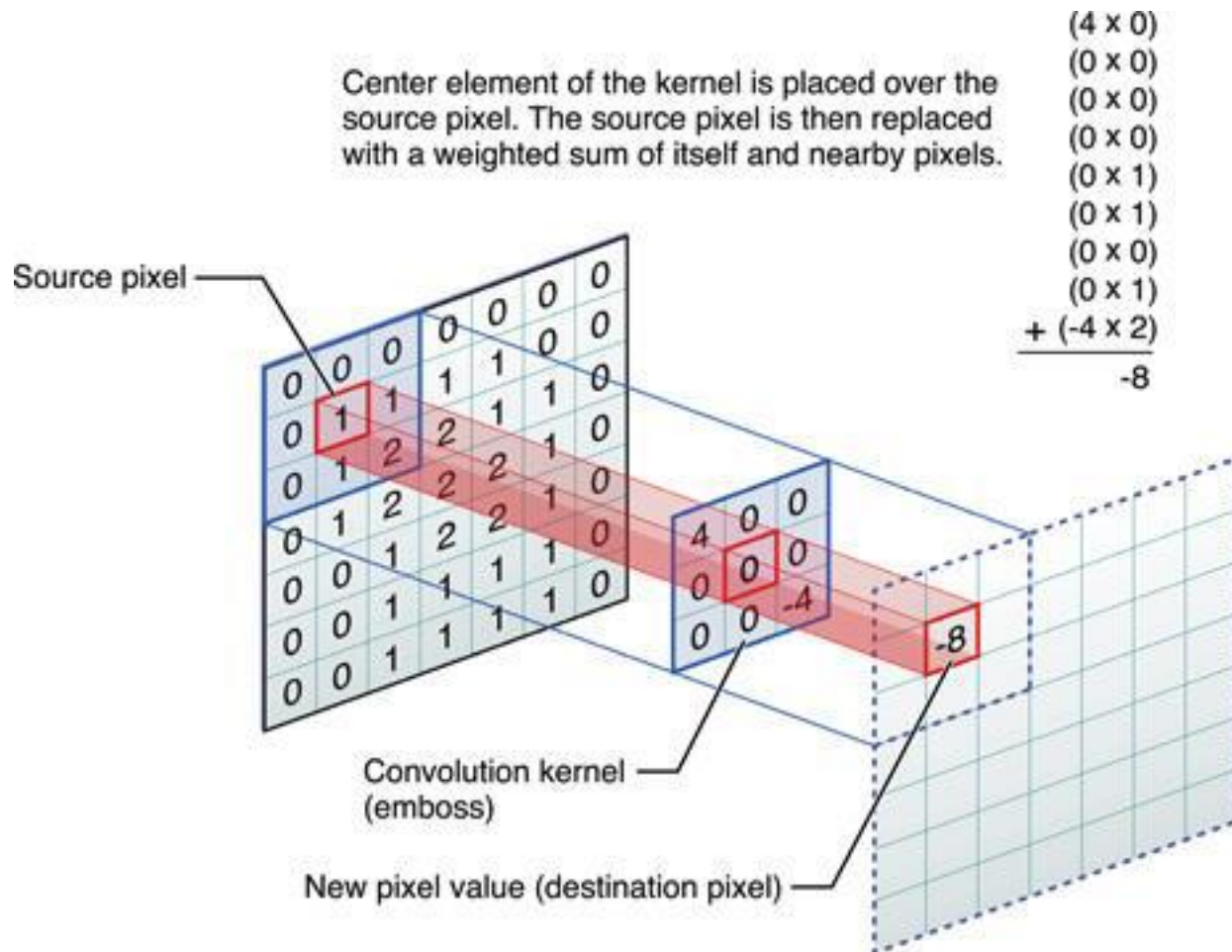


# Spatial Filtering

- Refers to operations with image pixel values in the neighborhood and corresponding values of a subimage having same dimensions as the neighborhood.
- Subimage is called, a **filter**, **mask**, **kernel**, or a **window**.
- Values in a filter is referred to as **coefficients**.
- Filtering can be performed in
  - spatial domain
  - frequency domain
- There are 2 main types of spatial domain filtering:
  - linear spatial filtering (convolution filter/mask/kernel)
  - nonlinear spatial filtering



# Linear Spatial Filtering



# Linear Spatial Filtering

Linear filtering of an image of size  $M \times N$  with a filter of size  $m \times n$  is given by:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t) f(x + s, y + t)$$

where

$a = (m-1)/2$  and  $b = (n-1)/2$

$w(-1, -1)$	$w(-1, 0)$	$w(-1, 1)$
$w(0, -1)$	$w(0, 0)$	$w(0, 1)$
$w(1, -1)$	$w(1, 0)$	$w(1, 1)$

Mask coefficients, showing  
coordinate arrangement

1	1	1
-1	2	1
-1	-1	1

Kernel

Example

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Image

Step-1

1	1	1		
-1	4	2	2	3
-1	-2	1	3	3
	2	2	1	2
	1	3	2	2

Input Image,  $f$



5			

output Image,  $g$

1	1	1
-1	2	1
-1	-1	1

Example

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Step-2

1	1	1	
-2	4	2	3
-2	-1	3	3
2	2	1	2
1	3	2	2



5	4		

Input Image,  $f$

output Image,  $g$

1	1	1
-1	2	1
-1	-1	1

Example

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Step-3

	1	1	1
2	-2	4	3
2	-1	-3	3
2	2	1	2
1	3	2	2



5	4	4	

Input Image,  $f$

output Image,  $g$

1	1	1
-1	2	1
-1	-1	1

Example

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Step-4

		1	1	1
2	2	-2	6	1
2	1	-3	-3	1
2	2	1	2	
1	3	2	2	



5	4	4	-2

Input Image,  $f$

output Image,  $g$

1	1	1
-1	2	1
-1	-1	1

Example

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Step-5

1	2	2	2	3
-1	4	1	3	3
-1	-2	2	1	2
	1	3	2	2



5	4	4	-2
9			

Input Image,  $f$

output Image,  $g$

# Smoothing Linear Filters

## 1. Averaging Filters

- The idea is to replace the value of every pixel by the *average* of the gray levels defined by the filter mask.
- Random noise consists of sharp transitions in gray levels.
  - So, the most obvious application is *noise reduction*.
- The undesirable effect is the *blurring* of edges

 $\frac{1}{9} \times$ 

1	1	1
1	1	1
1	1	1

(a)

 $\frac{1}{16} \times$ 

1	2	1
2	4	2
1	2	1

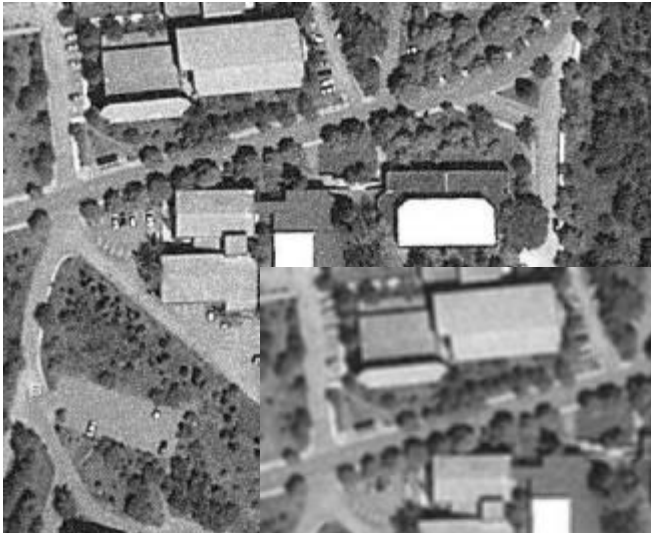
(b)

(a) Box filter

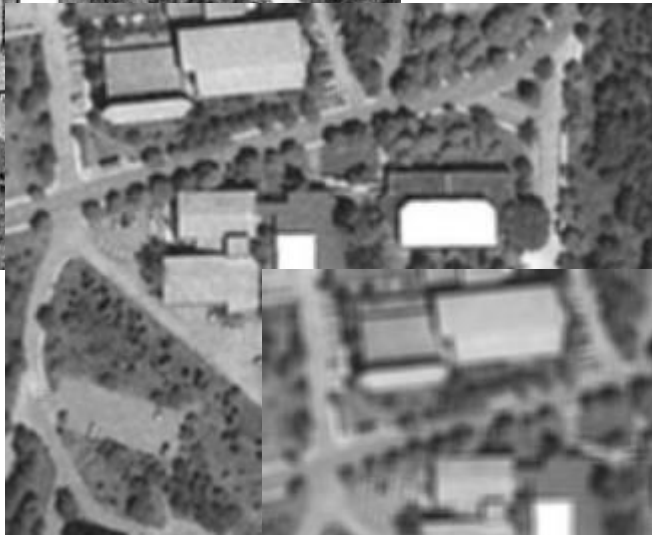
(b) Weighted average filter



# Effect of the filter size



- Original image



- 5x5 avg filter applied



- 11x11 avg filter applied

## 2. Gaussian smoothing

- Determine kernel coefficients using a 2D Gaussian distribution

=> **Center** pixel  
contributes the most  
=> **Moving away** from  
center decreases the  
contribution gradually

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

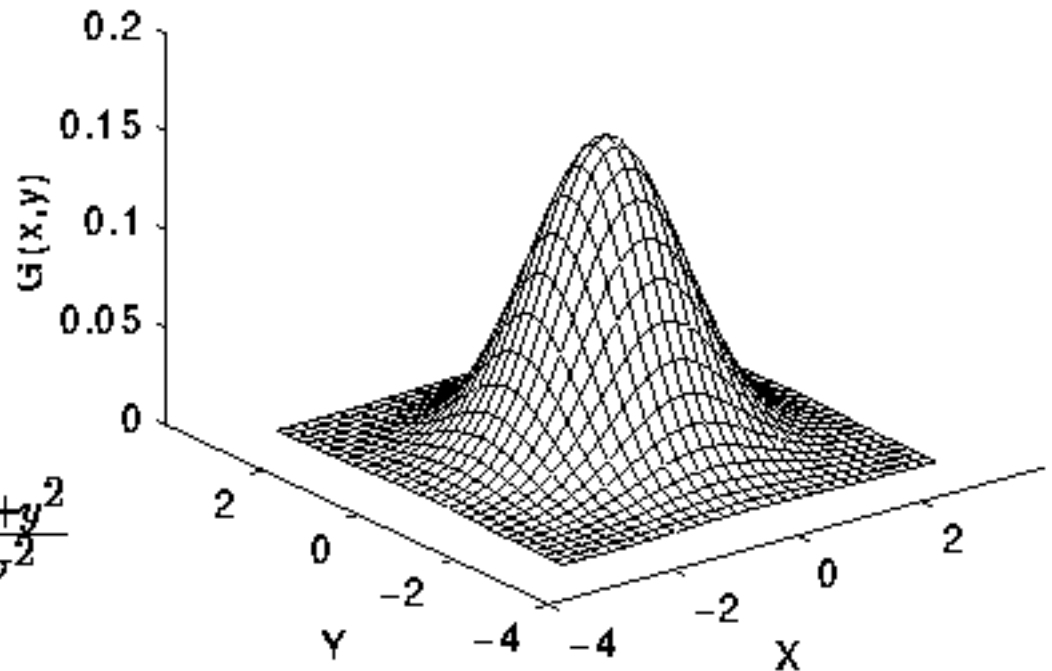


Figure 2 2-D Gaussian distribution with mean (0,0) and  $\sigma=1$

# Example: 5x5 Gaussian

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

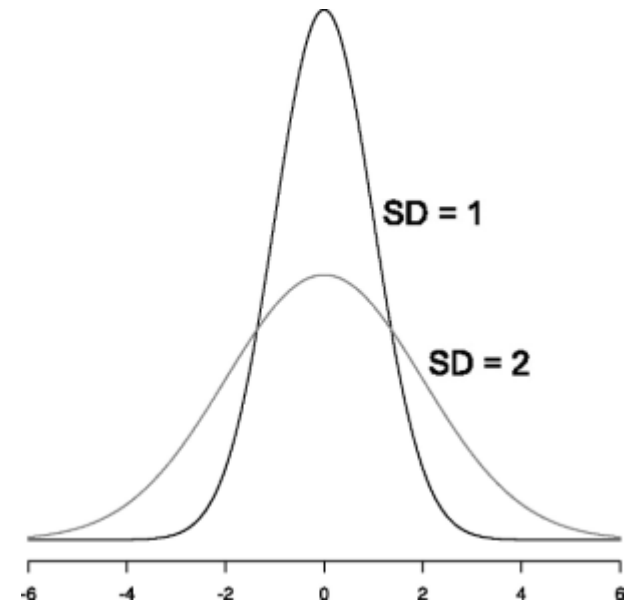
**Figure 3** Discrete approximation to Gaussian function with  $\sigma=1.0$

# Effect of the standard deviation

- If std dev is large enough
  - becomes an average filter
- What if std dev = 0?
  - becomes a unit impulse

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

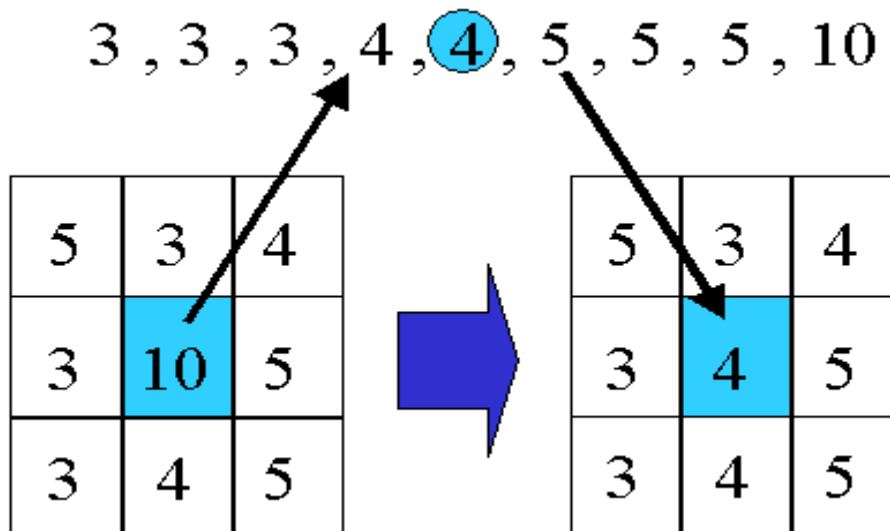
→ No effect on image



# Smoothing Non-linear Filters

- Most common: Median filter
  - Sort all pixel values inside kernel
  - Take the median and assign it to center pixel

An illustration with 3x3 filter:



# Median Filter: Example

- Good for removing salt&pepper noise



# Median Filter

- Less blurring when compared to average filter

3x3 average



3x3 median



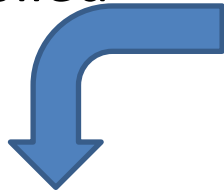
# Min and Max Filters

- After sorting pixels in ascending order
  - Take 1st value                   => Min Filter
  - Take the last                   => Max Filter
- Min Filter: Finds the local darkest point
  - Output image becomes darker
- Max Filter: Finds the local brightest point
  - Output image becomes brighter

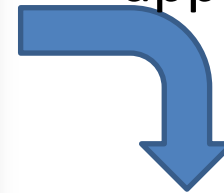


# Example: Min & Max Filters

5x5 **Min** filter  
applied



5x5 **Max** filter  
applied



# Sharpening Linear Filters

- Sharpening is the operation to highlight fine details or enhance the details that has been blurred
  - could be accomplished by *differentiation*
  - The *derivative* is defined in terms of *differences*
- 2 types of derivatives:
  - 1st order
  - 2nd order

# First order derivatives:

## Gradient filters

- First order derivative of a one dimensional function  $f(x)$  is:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- Prewitt filter: replace  $f(x)$  with  $f(x+1) - f(x-1)$

– Vertical

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

– Horizontal

$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

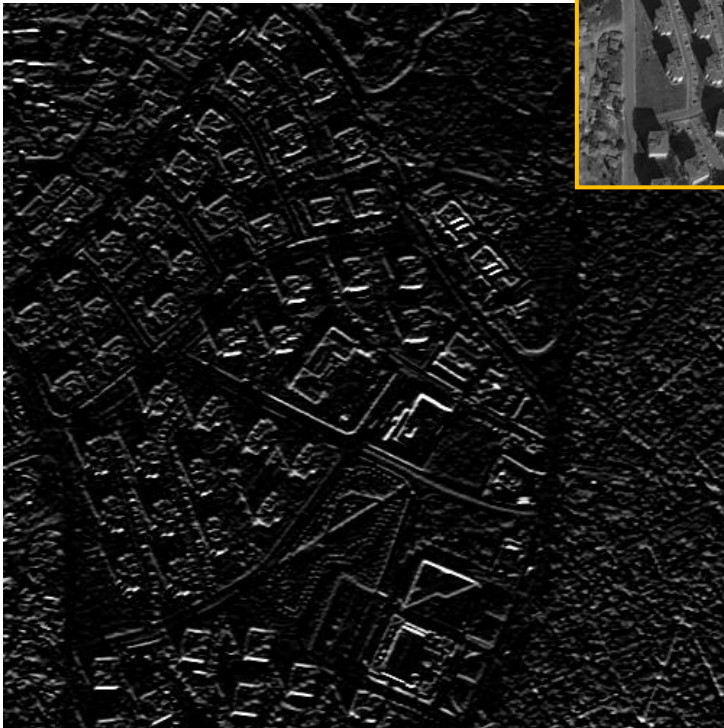
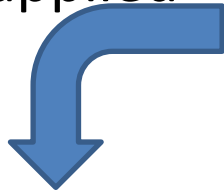
Generally, 3x3 versions are used:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

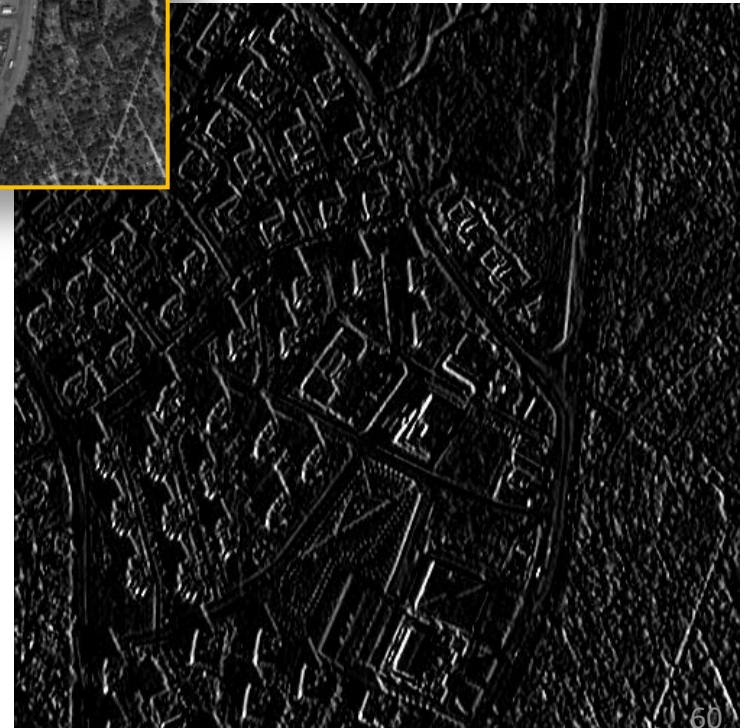
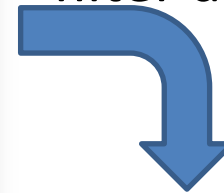
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

# Example: Prewitt Filters

Horizontal Prewitt  
filter applied



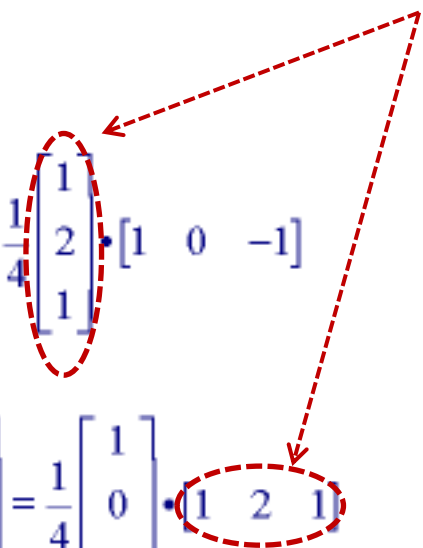
Vertical Prewitt  
filter applied



# First order derivatives

- Sobel filter:
  - a special version of Prewitt
  - takes the derivative in one direction smooths in the orthogonal direction

– Vertical

$$[\mathbf{h}_x] = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot [1 \ 0 \ -1]$$


– Horizontal

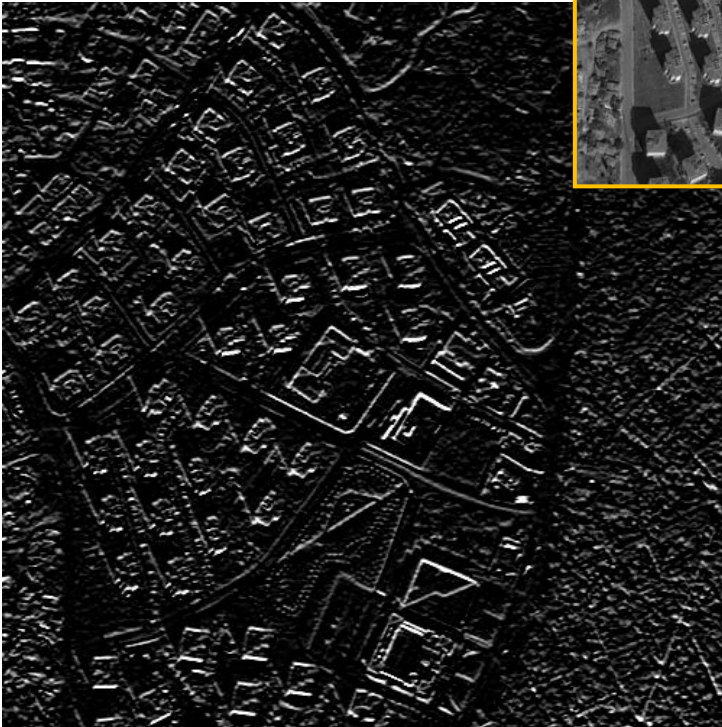
$$[\mathbf{h}_y] = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \cdot [1 \ 2 \ 1]$$

Value 2 gives more importance to *center*

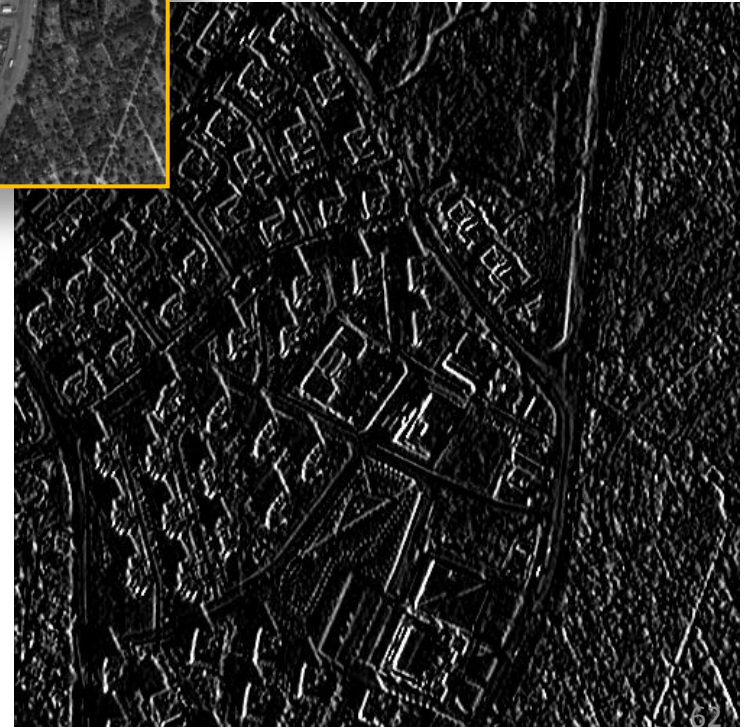
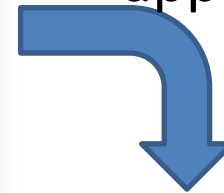


# Example: Sobel Filters

Horizontal Sobel  
filter applied



Vertical Sobel filter  
applied



# Second order derivatives

- Used to construct a Laplacian filter mask
- Laplacian is an *isotropic* filter where the response of the filter is independent of the direction of the discontinuities in the image
  - Isotropic filters are *rotation invariant*, which means that if you rotate and filter the image or if you filter and then rotate the image you get the same result

The operator is linear and can be expressed in x direction by:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

# Second order derivatives

The operator in y direction:

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

**Then, 2-D Laplacian is** sum of both

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$



# Laplacian Filters

considers x and y  
coordinates

Isotropic results  
for  $90^\circ$

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

considers x, y  
and 2 diagonal  
coordinates

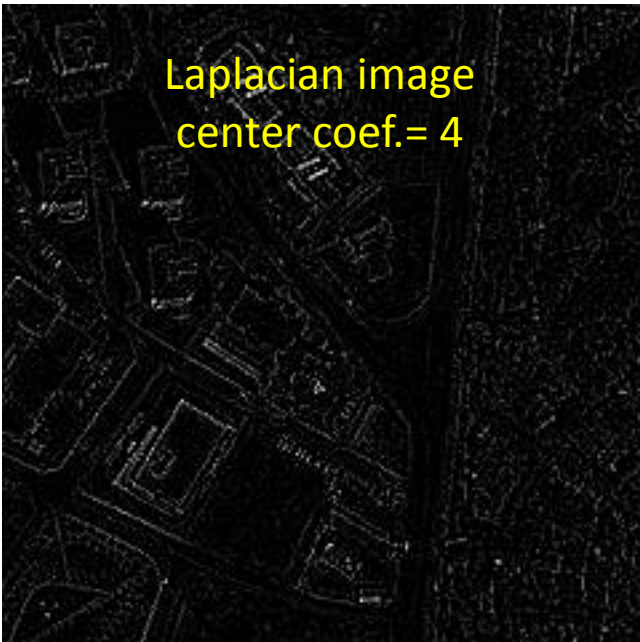
Isotropic results  
for  $45^\circ$

# Laplacian Filters

- Laplacian operator **highlights** gray level discontinuities and **de-emphasizes** the slowly varying gray-levels.
- Background features can be recovered and sharpening effect can be preserved **by adding** the Laplacian image to the original image
- Depending on the choice of the Laplacian coefficients the following criteria is used for enhancement:

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{•If the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{•If the center coefficient of the Laplacian mask is positive} \end{cases}$$

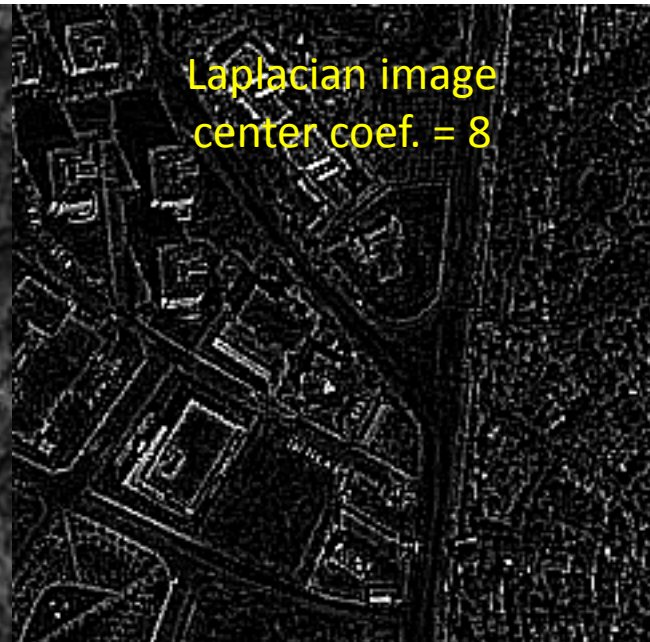
Laplacian image  
center coef.= 4



Original image



Laplacian image  
center coef. = 8



Enhanced images:  
Original + Laplacian



# References

- Read more:

Chapter 3: Digital Image Processing, 2nd Edition. Rafael C. Gonzalez. ISBN: 978-0201180756.

Chapter 7: Computer Processing of Remotely-Sensed Images: An Introduction, 3rd Edition. Paul M. Mather. ISBN: 978-0-470-02101-9.